International Journal of Modern Physics C,

SIMPLE AND EFFICIENT ALGORITHM FOR LARGE SCALE MOLECULAR DYNAMICS SIMULATION IN HARD DISK SYSTEM

MASAHARU ISOBE*
Department of Physics, Kyushu University 33
Fukuoka 812-8581, Japan

Received (received date)
Revised (revised date)

A simple and efficient algorithm of the molecular-dynamics simulation of the hard disk system based on the Event-Driven method is developed. From the analysis of algorithm, the complexity is $\mathcal{O}(\log N)$ per 1 event, and the constant coefficient of the complexity is smaller than conventional efficient algorithm based on the concept of Cell-Crossing Event. The maximum performance more than 460 millions of collisions per CPU-hour on the Alpha600 compatible in 2500 particle system is achieved. An extension to the infinite-space system based on this algorithm is also proposed.

1. Introduction

Molecular dynamics method (MD) was worked in a paper for the first time by Alder and Wainwright in 1957,[1] and there is a series of works after it by them.[2,3,4] Their simulations were performed with many hard disks (or hard spheres in 3-d) near the liquid-solid phase-transition point, and they found that the system was crystallized despite the particle had only repulsive force. These discover smashed the common sense of those days, and gave a big influence to the development of the study in the computer simulation.

In the hard disk system, the dynamics consists of only collisions and straight-line movement. Since distinct events occur one after another in time, we do not need to integrate the differential equations with a constant time step based on Newton's equation of motion. The method that is based on the finite constant time step and integration with the equations of particles step by step in time are sometimes called "Time-Step-Driven Molecular Dynamics" (TSDMD). On the other hand, in the hard disk system, the simulation that proceeds based on events is called "Event-Driven Molecular Dynamics" (EDMD). Compared with TSDMD simulation, the algorithm of EDMD simulation is completely different. We need the knowledge of an advanced algorithm and a data structure to perform the efficient simulation in EDMD. The strategy of direct computation of particle-pairs result in the complexity $\mathcal{O}(N^2)$ for large particle number N. The point of improvement in the speed in hard disk system is how we deal with the queue of a future event and the data structure

*E-mail: isobe@stat.phys.kyushu-u.ac.jp.

well.

The improvement of complexity in the algorithm of large-scale hard disk system was developed by Rapaport (1980).[5,6] This algorithm is based on the concept of sub-cell method,[7] and both Collision Event and Cell-Crossing Event are stored into Multiple-Event Time List. Then the minimum time is searched by Binary Search Tree(BST).[8] When the event occurs, the particle-pair or the particle - sub-cell respectively relevant to Collision Event or Cell-Crossing Event is deleted, and collision time for the particle relevant to the event is re-computed and new nodes are created. The BST is reconstructed by inserting these new nodes, and the minimum time is searched. On this algorithm, the averaged complexity per Event become $\mathcal{O}(\log N)$, and the reduction of the large amount of computation is realized.

However, since the algorithm of Rapaport is very complicated and difficult to understand, in the 90s the several algorithms to simplify a data structure and improve the efficiency in the large-scale molecular dynamics simulation were proposed.[9,10,11] Marín et al.[12] developed the technique of Local Minima Algorithm (LMA) to avoid additional re-computation for Event List. When we actually schedule future event list, LMA put only the minimum event time relevant to each particle into Complete Binary Tree (CBT). In 1995, Marín and Cordero[13] compared various $\mathcal{O}(\log N)$ searching algorithms actually in EDMD simulation, systematically. They concluded that the efficiency of Complete Binary Tree (CBT) was the most suitable for hard disks system in all density regions. In the number increase of particles, CBT was clearly showed that efficiency increased significantly from other searching algorithms. Compared the BST of Rapaport with CBT of Marín et al., although the complexity is the same order $\mathcal{O}(\log N)$, a simplicity, efficiency and memory reduction, and constant coefficient is different actually carrying out EDMD simulation.

In this paper, we developed an algorithm based on a strategy different from Cell-Crossing type. The algorithm is extended on Exclusive Particle Grid Method (EPGM) developed by Form et al.,[15] (Sec. 2) Then, bookkeeping method[16] is applied. (Sec. 3) Compared with the Cell-Crossing type, our algorithm extended the concept of Linked-Cell Method[17,18] and Neighbor List, which are often used in TSDMD to carry out an efficient simulation.[19] From the analysis of complexity, we show our algorithm is smaller than the complexity of Cell-Crossing type. By an empirical evaluation of the simulation in hard disk system, our code could be showed that the performance was better than that of any past-published works.

In either soft-core or hard-core system, it is regarded as the improvement in the speed base on conventional sub-cell method being impossible in the system with infinite volume.[10] This is because the space spreads out infinitely, infinite arrays must be prepared for infinite sub-cells. We developed the method of compressing information about the infinite sub-cells into limited finite array. In addition, the hashing method, which is known as the most efficient searching algorithm, is applied to our method in order to pull out the information on a neighbor cells in high speed. Especially, it is found that the hashing method could be applied easily on our method.

Various applications in a very wide range field will be possible by changing external field and the collision rule in the large-scale EDMD. The typical examples performed by EDMD so far are as follows; phase transition in the equilibrium system (solid-liquid transition and 2-dimensional melting),[1,2,3,4] the non-equilibrium fluid system (e.g. Rayleigh-Bénard convection),[20,21] the non-equilibrium chemical-reaction system,[22] the non-equilibrium dissipative system (granular system),[23,24,25] random disk packing.[26,27] When large-scale computation become possible in these system, the worth of simulation in a hard disk system must increase significantly.

The outline of this paper is as follows. In Sec. 2 and 3, the developed algorithms themselves are explained. The comparisons with the algorithm of Cell-Crossing type by analyzing the complexity are shown in Sec. 4. The empirical evaluation is also given in Sec. 5. The extension to the infinite system based on developed algorithms is explained in Sec. 6. In Sec. 7, a short summary and a final comment are presented.

2. Extended Exclusive Particle Grid Method

Sub-cell method is often used to achieve the significant reduction of computation. In the EDMD, the complexity of producing and updating future event list for restoring event time of each particle-pair are reduced to $\mathcal{O}(N)$ and $\mathcal{O}(1)$, respectively, by the sub-cell method. LCM is the method of dividing the system by small sub-cells. When the size of sub-cell is bigger than a particle diameter, we will have a difficulty to code a program, because we do not know how many particles enter in each sub-cell. Therefore, link lists must be prepared for the particles in each sub-cell.

On the other hand, another efficient sub-cell method, called Exclusive Particle Grid Method (EPGM), was individually developed by Buchholtz and Pöschel (1993)[14] and Form, Ito, and Kohring (1993)[15] to simulate the soft-core granular particle system in TSDMD. In this method, there is only one particle in each sub-cell. Though the EPGM is substantially the method of putting a particle to one sub-cell in LCM, it does not need to use pointers for neighbor sub-cells or link list. Here sub-cells in EPGM are called "grid". In the EPGM, the length of grid l_{gx} is determined by the following inequality;

$$\sigma < l_{gx} < \sqrt{2}\sigma, \tag{1}$$

where σ is the radidus of particle. One example of the number of grid n_{gx}, n_{gy} in the system length and the length of grid l_{gx}, l_{gy} can be calculated by the following equations.

$$l_{gx} = INT(l_x/(\sqrt{2}\sigma)) + 1 \tag{2}$$

$$l_{gy} = INT(l_y/(\sqrt{2}\sigma)) + 1 \tag{3}$$

$$n_{gx} = l_x/l_{gx} \tag{4}$$

$$n_{gy} = l_y/l_{gy}. \tag{5}$$

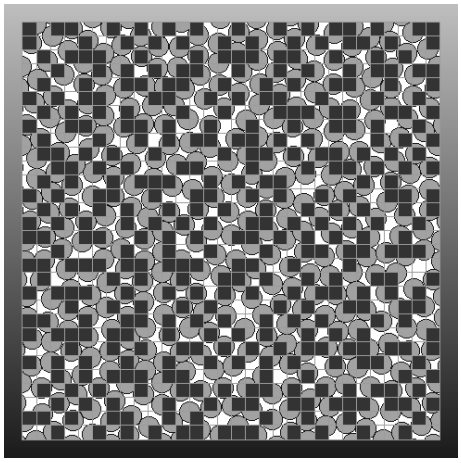

Figure 1: A typical example of the mapping pattern using EPGM when packing fraction $\nu = 0.70$. Both the position of hard disks and the occupied lattices are shown.

Note that the total number of grid is $n_g = n_{gx} \times n_{gy}$. As an analogy of the lattice spin system, EPGM is regarded as $N + 1$-states potts model of the square lattice system. This is because the particles are completely mapped into each lattice (i.e. one grid corresponds to one particle respectively), and we put 0 into the rest of grids in which a particle is not contained (see Fig. 1). Since continuous and random positions of the particles are mapped into the lattice, the specification of neighbor particles becomes very easy.
Form et al.[15] applied this algorithm in the high-density soft-core granular system with shot-distance interaction in TSDMD. They achieved high efficiency on a vector computer. Based on this algorithm, the extension of EPGM to EDMD in the hard disk system is developed.

When the system is in high-density, EPGM can be simply applied to EDMD. For a candidate of next colliding particle-pairs, we need to search only 24 neighbor grids, which form the square mask. If neighbor grid is not 0, the collision times of candidates of colliding particle-pairs are computed only registered particle number in the square mask. We call these 24 neighbor grids MIN, because this is the minimum mask in the simulation of EDMD. Note that if the smaller mask is used, the computation wills breakdown during the simulation, since a possibility of overlap between a central particle in the mask and particles out of the mask occurs. When EPGM is applied in the high-density system, the computation is optimized because a sufficient number of particles contained in the mask MIN. These are enough as candidates of particle-pairs of collision, and only required particles are registered in the mask MIN; the efficiency increases as a result of the computational reduction of collision time for neighbor particle-pairs.

On the other hand, when the system is in low-density, sufficient number of

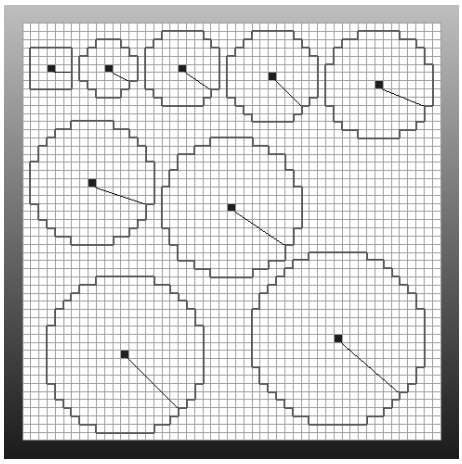

Figure 2: Minimum mask (MIN.) and larger masks, which are generated by the algorithm of making circle on the computer display, are shown. The solid lines in each mask denote the minimum distance from the central lattice of mask to the frame of mask.

particles as candidates of collision are not registered in the mask MIN. Under such a situation, the computation will breakdown, since collision time between the central particle in the mask and particles out of the mask will become next minimum time. In order to prevent this breakdown, the extension of EPGM is developed. The region must be extended to look for candidates of collision particle-pairs bigger than MIN. Since the rigorous isotropy of neighbors in EDMD are not necessarily demanded, the shape of mask might approximate with a rough circle. It is found that the mask approximated by the lattice-like grid with the circle can use the algorithm describing the circle on the discrete space of computer display. Figure 2 displayed the circles from R= 3 to R= 10 on the discrete space of computer display, which MIN is also showed over Fig.2. The total number of neighbor grids (mask) are 24 (MIN), 36 (R= 3), 68 (R=4), 96 (R=5), 136 (R=6), 176 (R=7), 224 (R=8), 292 (R=9), 348 (R=10), respectively. This extension of EPGM is called Extended Exclusive Particle Grid Method (EEPGM).

Compared with LCM, EEPGM is simple, because rough neighbor particles can be simply regulated by grids so that only the necessary minimum may be taken. In EEPGM, since link list and pointers for neighbor sub-cell is not necessary, a memory is also sharply reducible, and the number of operation for setting EEPGM is small, and the program become very simple. Moreover, the extension to the infinite volume system is easy when using a hashing method to EEPGM. This is explained at Sec. 6 in detail.

3. Neighbor List and Dynamical Upper Time Cut-Off

Though EEPGM realize significant reduction of computation compared with considering all particle-pairs, it is inadequate when large number of particle simulation is really performed. In this section, the next step of the strategy of increase in efficiency based on EEPGM is developed. Here, the concept of Neighbor List (NL)[16] is adopted. Since grid correspond to each particle, we can regard the registration of Neighbor List as already being completed. Therefore, we can only search neighbor particles along to the form of mask.

In the usual way of LCM+NL, after the system divided into sub-cells, particles are listed into link list. Then neighbor particles within radius r_{NL} are entered into Neighbor List from link list. However, since the length of both lists is unknown, they must be estimated by trial and error. Although registration of NL is completed by one operation in EEPGM, LCM+NL must use two different unknown size lists, which is accompanied by a complicated procedure and requires difficulty of programming. Therefore, EEPGM (+NL) is simple, which means that both debugging and extension do not require excessive effort, moreover, only the minimum nearest particles can be seen, because the system is divided into the minimum sub-cells, i.e. grid. Since registration of NL is completed by one operation in EEPGM, efficiency is better than LCM+NL at a result.

The next improvement in speed is that the computation of collision time only from particle-pairs which are registered in the mask of EEPGM during the time t_{NL}. Then the complexity of EEPGM for every event is reducible to $\mathcal{O}(1)$ instead of $\mathcal{O}(N)$. After time t_{NL} proceeds, the grids are again re-mapped in order to update neighbor particles. The time of Neighbor List t_{NL} must be determined that the central particle does not collides with the particle out of the mask completely. The far length of t_{NL} occurs a count mistake of particle-pairs, which produces negative collision time during the simulation. This conventional determination of t_{NL} needs huge trial and error. In order to overcome these difficulties, t_{NL} is determined by the following procedure. First, after completing EEPGM, the maximum velocity v_{max} in the system is searched, and the value of its velocity is restored (the complexity of this searching is the same order of EEPGM $\mathcal{O}(N)$). Next, time t_{NLmin} of the system is calculated. In this calculation we suppose both the central particle and the particle out of the mask has the maximum velocity and those particles undergoes head-on collision. If $t_{NL} = t_{NLmin}$, a count mistake of collision pairs in the system never occurred during the time t_{NLmin}. The minimum NL distance r_{NLmin} is required when t_{NLmin} is calculated, which become clear from the geometry of the adopted mask shown in Fig. 2. Therefore, t_{NLmin} is given by

$$t_{NLmin} = \frac{r_{NLmin} - (2\sigma_{max})}{2v_{max}} \tag{6}$$

where σ_{max} is the maximum radius of particle in the system. The minimum distances r_{min} for each mask are shown in Table 1.

When we simulate in the equilibrium system, this strategy will work well because t_{NLmin} hardly changes. However, in the non-equilibrium system (e.g. the relaxing

Table 1. The minimum distance for each mask.

	MIN.	R=3	R=4	R=5	R=6	R=7	R=8	R=9	R=10
r_{min}/l_{gx}	2	$\sqrt{5}$	$\sqrt{13}$	$\sqrt{18}$	$\sqrt{29}$	$\sqrt{40}$	$\sqrt{52}$	$\sqrt{72}$	$\sqrt{85}$

process or the system with heat bath) it breakdown because the maximum velocity changes drastically at every step. To overcome this difficulty, we must check the maximum velocity for each event with energy increase. Fortunately the complexity of this checking process is $\mathcal{O}(1)$. Therefore, in the simulation of the non-equilibrium system t_{NL} will change one after another. We call this changing t_{NL} techniques Dynamical Upper Time Cut-off (DUTC). The development of DUTC, EEPGM became applicable in the non-equilibrium system.

Although in the high-density system we do not need to update the grid pattern for a long time, in the low density we should often update the grid pattern when we use the mask MIN because the grid pattern changes drastically. In order to reduce the frequency of updating grid pattern, we can only use a bigger mask.

4. Analysis of Complexity

Analysis of complexity is one of important factor to estimate the efficiency of algorithm. In this section, a comparison of analysis of complexity between the algorithm of the Cell-Crossing type and the EEPGM + DUTC is shown. The difference between the algorithm of Cell-Crossing type and the strategy of EEPGM + DUTC is Cell-Crossing Event itself. Therefore, especially looking at this point, both complexities with a constant coefficient k are estimated in the case of $A \times N$ collisions being actually simulated. Note that the particle number N is supposed to be a quite large number and the techniques of improvement in the speed are also used in both algorithm denoted by Marín et al.[12]

- Cell-Crossing type (LCM + Cell-Crossing Event)
  - The initial and last step ($\times 1$)
    Linked-Cell Method — $k_{LCM} \times N$
    Computation of Event Time — $k_{PP} \times 9 \times N_c \times N + k_{PC} \times 4 \times N$
    Construction of Complete Binary Tree — $k_{CBT} \times N$
    Update the final position of particles — $k_{UPDATE} \times N$
  - Iteration Step (loop)
    $(A \times N) \times (k_{PP} \times 9 \times N_c + k_{CBT} \times \log N) + (B \times N) \times (k_{PC} \times 3 + k_{CBT} \times \log N)$

  where N_c is the number of particles contained in per sub-cell. The most important point is that the Cell-Crossing Event occurs by a certain ratio of Collision Event. Therefore, the additional computation of $B \times N$ times Cell-Crossing Events are needed, when we want to simulate $A \times N$ times Collision

Events. Since the complexity of the terms related to Cell-Crossing Event is always $\mathcal{O}(N)$, this is not negligible in the actual simulation.

- EEPGM + DUTC
  - Update of EEPGM ($\times C$)
    EEPGM — $k_{EEPGM} \times N$
    Computation of Event Time — $k_{PP} \times N_g \times N$
    Search the Maximum Velocity — $k_{MVS} \times N$
    Construction of Complete Binary Tree — $k_{CBT} \times N$
    Update the final position of particles — $k_{UPDATE} \times N$,
  - Iteration Step (loop)
    $(A \times N) \times (k_{PP} \times N_g + k_{CBT} \times \log N)$

  where N_g is the averaged particle number of the mask. Actually the value of C is an order $C \sim A$, which is the same order of the frequency of updating EEPGM.

Now, the comparison with order N in both algorithm with constant coefficient is as follows:

- Cell-Crossing type

  $(k_{LCM} + k_{PP} \times 9 \times N_c \times (A+1) + k_{PC} \times (3 \times B + 1) + k_{CBT} + k_{UPDATE}) \times N + (A \times k_{CBT} + B \times k_{CBT}) \times N \times \log N$

- EEPGM + DUTC

  $(k_{EEPGM} + k_{PP} \times 2 \times N_g + k_{MVS} + k_{CBT} + k_{UPDATE}) \times C \times N + (A \times k_{CBT}) \times N \times \log N$

The most striking point is that the complexity of Cell-Crossing Event is of order of $\mathcal{O}(N \log N)$ (i.e. $B \times k_{CBT} \times N \times \log N$). This result of analysis suggest that the efficiency of EEPGM + DUTC is better than Cell-Crossing type when the simulation with the enormous number of particle is performed. On the other hand, in the comparatively small particle system, the coefficient of $C \times N$ terms in EEPGM + DUTC may be larger than Cell-Crossing type. However, the difference might be quite small, and it is impossible to estimate an exact coefficient of algorithms analytically. To the author's knowledge, the coefficient of N terms is strongly dependent on the ratio of N_c to N_g, and the rough estimation shows both algorithms are same when $N_g/N_c \sim 4$. Actually, the increase of computing Cell-Crossing Events is almost not effect to CPU time as far as the simulation in the very large number particle system is actually performed.

5. Empirical Evaluation

At the stage of actually running the simulation, the order of the complexity is not so reliable, because it depends strongly on a constant coefficient when the number of the particle is relatively small. Moreover, the efficiency of the code changes significantly by the ability of a computer, the language, the performance of compiler, and the ability of programmers. Though a perfect comparison of efficiency of codes developed by the past workers is impossible, some literatures showed how many particle collisions per CPU-hour can their codes be computed by their computers.

Marín et al.(1993)[12] simulated with hard disk system, and Marín et al. also compared their code with the codes based on two main high-speed algorithms proposed by Rapaport(1980)[5] and Lubachevsky(1991),[9] respectively. At a result, it was shown that the efficiency of the code of Marín et al. was superior in the entire density region. Therefore, the author's code should be compared with the code of Marín et al., and it computed with the number of the particles equal. Marín et al. achieved the maximum performance at 16.07 millions of collisions per one CPU hour on workstation SUN690. Note that the efficiency showed only the highest performance because the performance is different for a different density of the system. On the other hand, author's code achieved the maximum performance 460 millions of collisions per one CPU hour (Alpha600 compatible, DEC-Fortran). It was found that the high efficiency was realized as for the author's code even if the performance of the machine was deducted.

Note that the workstation of our laboratory could actually simulate the 2,500,000 particle system. In this simulation, the amount of an installed memory was $250Mbyte$ and the computation performance was 130 millions of collisions per one CPU hour.

6. Extension to Infinite Volume System

In this section, a simple example in the open boundary system that does not have a ceiling is considered. This is the case that there is an energy source at the bottom of the system under the uniform gravity.

The system is also divided into grids by EEPGM. However, because the top of the system is opened, grid goes to the top of the system infinitely. This means that the number of arrays for grid becomes infinity, and the simulation is impossible from finite memory. To overcome this difficulty, the hashing method, which is well known as the fastest searching algorithm, is applied to keep the number of arrays in finite size and to simulate the dynamics of the system in high-efficiency.

- Construction of Data Structure

  Firstly, the serial number of grid $N_G(i_g, j_g)$ is defined by

$$N_G = N_{gx}(j_g - 1) + i_g, \tag{7}$$

  where N_{gx} and $i_g(1 \leq i_g \leq N_{gx})$ are the total number of grid and the index of grid in the horizontal direction, respectively; $j_g(1 \leq j_g \leq \infty)$ means the index of grid in the vertical direction. In addition, the maximum number of

the serial grid N_{Gmax} is calculated by $N_{Gmax} = N_{gx}(j_{gmax} - 1) + i_{gmax}$; the pair (i_{gmax}, j_{gmax}) is at the maximum grid pair of containing the particle.

The serial grid $N_G (1 \leq N_G \leq N_{Gmax})$ is one-dimensional array, in which particle number or 0 are listed. This is called one-dimensional *Virtual Array*. When j_{max} is large value, there are many 0's in the one-dimensional *Virtual Array*. Though there is only information that a particle is not just in grid, these 0 relates to the memory capacity directly. Now *Virtual Array* is compressed. After all, the necessary information is the particle number and its index of grid. Therefore, one-dimensional integer arrays are prepared for $A(N), BX(N), BY(N)$, and grids of 0 in *Virtual Array* are ignored, and then packed in order from the end; $A(N)$ stores the particle numbers only, and $BX(N)$ and $BY(N)$ stores indexes of i_g and j_g for each particle, respectively. If you want to know whether there is a particle in grid (i'_g, j'_g), you have only to search the index-pairs correspond to (i'_g, j'_g) in the lists of $BX(N), BY(N)$ linearly. However, this process is inefficient because the complexity of $\mathcal{O}(N)$ is necessary in searching.

Therefore, the hashing method known as an algorithm which can realize the searching with $\mathcal{O}(1)$ is applied. The following simplest hashing function is explained here as an example though various hashing functions can be considered and the room of the development is still left.

First, hashing function is defined by

$$k = [\frac{N_G - 1}{L}] + 1 \tag{8}$$

which means that *Virtual Array* is equally divided by the length L (e.g. $5 \sim 10$) and key k is calculated correspond to serial number N_G. Then, $k_{max} = [\frac{N_{Gmax}}{L}]$ is calculated using N_{Gmax}. Then additional arrays $C(k_{max}), D(k_{max})$ are prepared, these arrays are restored in $C(k)$ where k begins in $A(N)$ and in $D(k)$ how much size of arrays for each k. In this case, necessary arrays are $A(N), BX(N), BY(N), C(k_{max}), D(k_{max})$, and these are confined to a finite value. Since additional arrays to use the hashing method is only $C(k_{max}), D(k_{max}) (k_{max} < N)$, the amount of use of a memory slightly increase in comparison with the linear searching.

- Searching Process

  In order to know what the particle number is in the grid (i'_g, j'_g), the following process is carried out. First, a serial number is calculated by $N'_G = N_{gx}(j'_g - 1) + i'_g$. Next, k' is calculated by the equation of hashing function (8). Then, the searching ranges of BX, BY are limited only to index of $[C(k') \sim C(k') + D(k') - 1]$ $(\leq L)$. If there are equal pairs of grid (i'_g, j'_g) as a result of the searching of BX, BY, the index s of BX, BY reveals $A(s)$ which is the particle number in the grid (i'_g, j'_g). When equal pair is not found in BX, BY, there is no particle in the grid.

This procedure becomes possible in high-speed simulation. The complexity becomes $\mathcal{O}(1)$ instead of the linear searching $\mathcal{O}(N)$, because a searching is only carried out on the length of L with hashing method.

A computation is possible for other boundary conditions with the same procedure if one-dimensional *Virtual Array* can be created. Therefore, high-speed simulation is possible what kind of boundary is applied, in principle. There will be a room of the improvement in hashing function, which is the easiest dividing equally, because it is obviously inappropriate when particles are distributed heterogeneously.

The strategy of EEPGM has an advantage that the extension is easy. The ease of a development is an important factor of the evaluation of an algorithm.

Note that a problem in low-density system is that the arrays assigned to grid increases overwhelmingly in comparison with the number of the particles. This is not desirable from the point of the memory capacity as the same of infinite system. However, arrays for grid are compressed to the size of the particle number in the same way as described above. Since supplementary arrays is made by hashing method and information on neighboring grids is efficiently obtained, there is no problem both efficiency and memory capacity.

7. Concluding remarks

In this paper, we developed an algorithm for hard disk system without using Cell-Crossing Event, which is simple, efficient and easy to extend. EEPGM can be easy to extension because of its simplicity, which can never be realized in LCM. One example is the system that hard disks with various size of diameter coexist. Though there was a limitation in the degree of the poly-dispersion with EPGM described to Buchholtz et al.,[14] EEPGM can be applied easily to those system. First, grid is made based on the smallest particle radius in the system. Next, we have only to check the nearest grids by using the suitable mask of the bigger level when the poly-dispersity increases. Like this, EEPGM has a wider application than EPGM.

This code achieved the fastest simulation speed in the world; about 460 million of collisions per CPU hour for the 2500 disk system on the VT-Alpha-600. Since the order of complexity is $\mathcal{O}(\log N)$, the increase of complexity is slow when the particle number increase. Now, we can carry out large-scale molecular dynamics simulation ($\sim 10^6$) on the usual Workstation in our laboratory.

Hard particle simulation is a powerful tool to study the fluidized state described by the kinetic theory or hydrodynamics. Therefore, the large-scale simulation in the hard particle system will become an important subject.

At the end, the algorithm in this paper is suitable for the scalar machine, and the development of the algorithm for the parallel machine is a future subject. Note that we discuss only hard disks system in 2-d for simplicity, but an extension to hard spheres system in 3-d is easy to be carried out.

Acknowledgements

The auther thanks Prof. Nakanishi for reading the manuscript and making a number of helpful suggestions. He also gratefully acknowledge helpful discussions with Dr. Muranaka on several points in the paper. A part of the computation in this work was done by the facilities of the Supercomputer Center, Institute for Solid State Physics, University of Tokyo.

References

1. B. J. Alder and T. E. Wainright, *J. Chem. Phys.* **27**, 1208 (1957).
2. B. J. Alder and T. E. Wainright, *J. Chem. Phys.* **31**, 459 (1959).
3. B. J. Alder and T. E. Wainright, *J. Chem. Phys.* **33**, 1439 (1960).
4. B. J. Alder and T. E. Wainright, *Phys. Rev.* **127**, 359 (1962).
5. D. C. Rapaport, *J. Comput. Phys.* **34**, 184 (1980).
6. D. C. Rapaport, *The art of molecular dynamics simulation* (Cambridge University Press, Cambridge, 1995).
7. J. J. Erpenbeck and W. W. Wood, in *Modern Theoretical Chemistry Vol.6, Statistical Mechanics Part B*, ed. B.J.Berne (Plenum, New York, 1977), Chap. 1, p. 1.
8. D. E. Knuth, *Sorting and Searching, The Art of Computer Programming*, Vol.3, (Addison-Wesley, Reading, Mass., 1973).
9. B. D. Lubachevsky, *J. Comput. Phys.* **94**, 255 (1991).
10. K. Shida and Y. Anzai, *Comput. Phys. Commun.* **69**, 317 (1992).
11. A. T. Krantz, *ACM Trans. Model. Comput. Sim.* **6**, 185 (1996).
12. M. Marín, D. Risso, and P. Cordero, *J. Comput. Phys.* **109**, 306 (1993).
13. M. Marín and P. Cordero, *Comput. Phys. Commun.* **92**, 214 (1995).
14. V. Buchholtz and T. Pöschel, *Int. J. Mod. Phys.* **C4**, 1049 (1993).
15. W. Form, N. Ito, and G. A. Kohring, *Int. J. Mod. Phys.* **C4**, 1085 (1993).
16. L. Verlet, *Phys. Rev.* **159**, 98 (1967).
17. B. Quentrec and C. Brot, *J. Comput. Phys.* **13**, 430 (1975).
18. R. W. Hockney and J. W. Eastwood, *Computer Simulation Using Particles* (McGraw-Hill, New York, 1981).
19. M. P. Allen and D. J. Tildesley, *Computer Simulation of Liquids* (Clarendon Press, Oxford, 1987).
20. M. Mareschal, M. Malek-Mansour, A. Puhl, and E. Kestemont, *Phys. Rev. Lett.* **41**, 2550 (1988).
21. D. C. Rapaport, *Phys. Rev. Lett.* **60**, 2480 (1988).
22. T. Kawakatsu and A.Ueda, *J. Phys. Soc. Jpn.* **56**, 847 (1987).
23. I. Goldhirsch and G. Zanetti, *Phys. Rev. Lett.* **70**, 1619 (1993).
24. S. McNamara and W. R. Young, *Phys. Rev.* **E53**, 5089 (1996).
25. C. Bizon, M. D. Shattuck, J. B. Swift, W. D. McCormick, and H. L. Swinney, *Phys. Rev. Lett.* **80**, 57 (1998).
26. B. D. Lubachevsky and F. H. Stillinger, *J. Stat. Phys.* **60**, 561 (1990).
27. B. D. Lubachevsky, F. H. Stillinger, and E. N. Pinson, *J. Stat. Phys.* **64**, 501 (1991).